JOURNAL OF COMPUTER AND SYSTEM SCIENCES 48, 231–254 (1994)

Exact Lower Time Bounds for Computing Boolean Functions on CREW PRAMs*

MARTIN DIETZFELBINGER†

Fachbereich Informatik, Universität Dortmund, D-44221 Dortmund, Germany

MIROSŁAW KUTYŁOWSKI‡

Instytut Informatyki, University of Wrocław, PL-51-151 Wrocław, Poland

AND

RÜDIGER REISCHUK§

Institut für Theoretische Informatik, Technische Hochschule Darmstadt, D-64283 Darmstadt, Germany

Received August 2, 1991; revised October 20, 1992

The time complexity of Boolean functions on abstract concurrent-read exclusive-write parallel random access machines (CREW PRAMs) is considered. We improve results of Cook, Dwork, and Reischuk (*SIAM J. Comput.* **15** (1986), 87–97), and extend work of Kutyłowski (*SIAM J. Comput.* **20** (1991), 824–833), who proved a lower time bound for the OR function on such machines that equals the upper bound. We provide a general means for obtaining exact (i.e., correct up to an additive constant) lower bounds, which works for many Boolean functions, in particular all symmetric functions. The new approach is based on the fact that Boolean functions can be represented as polynomials with integer coefficients and that the degree of such a polynomial can be taken as a complexity measure. For some functions, e.g., AND and PARITY, the exact time bound also holds for nondeterministic machines. For probabilistic machines, we obtain exact lower time bounds for PARITY in the unbounded error model and, utilizing results by Szegedy (Ph.D. dissertation, University of Chicago, 1989), prove a general lower bound valid for all Boolean functions in the bounded error model. We further show that the (bounded error) probabilistic time complexity of Boolean functions on CREW PRAMs differs at most by a constant factor from the

---

* A first version of this paper was presented at the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures, Crete, July 1990.

† Partially supported by DFG Grant ME 872/1-4 and by DFG-Forschergruppe "Effiziente Nutzung massiv paralleler Systeme, Teilprojekt 4." This author was affiliated with the Universität–Gesamthochschule–Paderborn, Germany while this paper was written.

‡ Part of this work was done while visiting the TH Darmstadt supported by the Alexander-von-Humboldt-Stiftung, later supported by program RP.I.09 sponsored by the Polish government.

§ Partially supported by the International Computer Science Institute, Berkeley, California.

0022-0000/94 $6.00

deterministic time complexity. We also obtain exact bounds for machines that allow a few processors to try to write to the same cell simultaneously. These bounds are stronger than those which follow automatically from the exclusive-write bounds. No tight bounds for this model were known before.

1. INTRODUCTION

This paper studies the time complexity of Boolean functions on abstract concurrent-read exclusive-write parallel random access machines (CREW PRAMs). A PRAM consists of processors $P_1, P_2, \ldots$ and a common random access memory consisting of cells $C_1, C_2, \ldots$. The n bits of the input are given in the first n cells; at the end of the computation the output is the content of the first memory cell. We are mainly interested in the CREW model, in which several processors may simultaneously read a common memory cell, but never more than one processor is allowed to simultaneously write to a common memory cell. We assume that the number of processors active in the computation and the type of operations they perform, as well as the size of the words that might be stored in a memory cell, are unrestricted. We will also consider variants of this model, viz., nondeterministic and probabilistic versions. (For precise definitions see Sections 2.2, 5, and 6.)

Recently, substantial progress has been made in understanding the time complexity of Boolean functions, i.e., functions $f: \{0,1\}^n \to \{0,1\}$, on abstract CREW PRAMs. The most significant steps in this development were the lower bound by Cook, Dwork, and Reischuk [7], Nisan's [18] characterization of the time complexity of Boolean functions on CREW PRAMs up to a constant factor, and finally Kutyłowski's [13] result that determines the CREW complexity of many Boolean functions, including the OR of n bits, up to an *additive* term between zero and one. In this paper the last approach, which yields exact time bounds, is developed further. Let us begin with a closer look at the mentioned results.

DEFINITION 1.1. (a) The Fibonacci numbers F_k, $k \geqslant 0$, are defined by $F_0 = 0$, $F_1 = 1$, and $F_k = F_{k-2} + F_{k-1}$ for $k \geqslant 2$.

(b) (The "Fibonacci bound") $\phi(x) := \min\{t : F_{2t+1} \geqslant x\}$.

Note. $\log_b x \leqslant \phi(x) \leqslant \log_b x + 1.34$, for $x \geqslant 1$, where $b = [\frac{1}{2}(1+\sqrt{5})]^2 = 2.618\ldots$. We have $\log_b x = c \log x$,[1] for $c = 0.7202\ldots$.

Notation. In the following, $x_1, \ldots, x_n$ denote variables, i.e., literals, and $\mathbf{a} = (a_1, \ldots, a_n)$, $\mathbf{b} = (b_1, \ldots, b_n)$ denote elements of $\{0,1\}^n$.

DEFINITION 1.2. (a) $\mathbf{B}_n$ denotes the set of all Boolean functions of n variables, i.e., functions $f: \{0,1\}^n \to \{0,1\}$.

(b) CREW(f) denotes the minimal number of steps a CREW PRAM needs to compute $f \in \mathbf{B}_n$.

[1] Throughout the paper, log stands for log to the base 2.

Recall that a Boolean decision tree for n variables is a binary tree in which each internal node has two sons and is labeled by one of the variables, and each leaf is labeled by 0 or 1. Given such a tree and an input $(a_1, ..., a_n) \in \{0, 1\}^n$, we can perform a computation in the following way: starting from the root of the tree we go downwards in the tree; at each internal node encountered we test the input bit that corresponds to the variable that labels the node; if it is 0, then we go to the left son; otherwise we go to the right son. The label of the leaf that is eventually reached is the result of the computation.

DEFINITION 1.3. $D(f)$ denotes the minimal depth of a Boolean decision tree for n variables that computes $f \in \mathbf{B}_n$.

Cook, Dwork, and Reischuk showed that the OR of n inputs, denoted by OR_n, can be computed on an n-processor CREW PRAM faster than in the obvious binary tree fashion, which takes $\lceil \log n + 1 \rceil$ steps [31, p. 363]. If $n \cdot 2^n$ many processors are provided, the bound applies to all Boolean functions. The general upper bound can also be be formulated in terms of the decision tree complexity.

THEOREM 1.4. (a) $\mathrm{CREW}(\mathrm{OR}_n) \leqslant \phi(n) \leqslant 0.73 \log n + 1.34$.

(b) $\mathrm{CREW}(f) \leqslant \phi(n) + 1$, *for arbitrary* $f \in \mathbf{B}_n$.

(c) $\mathrm{CREW}(f) \leqslant \phi(D(f)) + 1$, *for arbitrary* $f \in \mathbf{B}_n$.

(For (a) see [7]; for (b) see [7, 13]; the idea for proving (c) is to test in parallel each path in the decision tree by applying the algorithm for computing the OR for each path. In this way, the unique path that describes the computation in the decision tree is found.)

Remark. A comment on the significance of upper bound results for the abstract CREW PRAM may be in place. In principle, this model is meant for proving lower bounds. Because there are no restrictions on the number and the computational power of the processors, the lower bounds proved for this model apply to all more realistic variants of the CREW PRAM. If we give upper bounds (like in the preceding theorem), this only indicates the limits to the lower bounds that can be proven within the general model; such upper bounds are usually not intended as giving realistic or efficient algorithms. In the present paper, a series of lower bounds is proved that differ from the upper bounds only by a small additive constant. Such an "exact" lower bound tells us that no larger lower bounds can be proved within the abstract CREW PRAM model. It does not necessarily determine the complexity of the computational problem at hand on more realistic variants of the CREW PRAM model. On the other hand, we note that many important Boolean functions can be computed in time close to $\phi(n)$ on a CREW PRAM even if only a realistic number of processors (polynomial or linear in n) and memory cells that can store only reasonably sized words (constant or logarithmic in n) are available [8].

In [7], a lower bound $\Omega(\log n)$ for CREW(OR_n) was proved. For other Boolean functions, the bound can best be expressed in terms of the *critical complexity* $c(f)$ of f, also known as the *sensitivity* of f. (We follow the notation from [31].)

DEFINITION 1.5. $c(f) := \max\{|\{i: f(\mathbf{a}^{\{i\}}) \neq f(\mathbf{a})\}| : \mathbf{a} \in \{0,1\}^n\}$, where $\mathbf{a}^{\{i\}} = (a_1, ..., \bar{a}_i, ..., a_n)$ for $\mathbf{a} \in \{0,1\}^n$ and $1 \leqslant i \leqslant n$.

THEOREM 1.6 ([7]; improved in [20]). $\mathrm{CREW}(f) \geqslant \log_4(c(f))$, *for all* $f \in \mathbf{B}_n$. *In particular,* $\mathrm{CREW}(\mathrm{OR}_n) \geqslant \log_4 n = \frac{1}{2}\log n$.

Nisan settled the question about the asymptotic CREW complexity of Boolean functions. A key step in his proof was generalizing $c(f)$ to the *block-critical complexity* $\mathrm{bc}(f)$ (also known as *block sensitivity*). He also relates CREW(f) to the sensitive complexity $s(f)$, which is a standard complexity measure (it is sometimes called *certificate complexity*).

DEFINITION 1.7. (a) $\mathrm{bc}(f)$ is the maximum of the numbers

$$\max\{l: \exists_{S_1, ..., S_l \subseteq \{1, ..., n\}, \text{ disjoint}} f(\mathbf{a}^{S_j}) \neq f(\mathbf{a}),\ 1 \leqslant j \leqslant l\},$$

taken over all inputs $\mathbf{a}$, where $\mathbf{a}^S$ is obtained from $\mathbf{a}$ by negating all bits in positions $i \in S$.

(b) $s(f)$ is the maximum of the numbers

$$\min\{k: \exists_{S \subseteq \{1, ..., n\}, |S| = k} f(\mathbf{a}) = f(\mathbf{b}) \text{ if } \forall_{i \in S}\, a_i = b_i\}$$

taken over all inputs $\mathbf{a}$.

THEOREM 1.8 [18]. *For all* $f \in \mathbf{B}_n$ *the following holds*:

$$\mathrm{CREW}(f) = \Theta(\log(D(f))) = \Theta(\log(s(f))) = \Theta(\log(\mathrm{bc}(f))).$$

Although Nisan's result determined the CREW complexity of Boolean functions up to a small multiplicative constant, it did not close the gap between the upper bound $0.73 \log n$ of Theorem 1.4 and the lower bound $\frac{1}{2}\log n$ of Theorem 1.6 for functions like OR_n. This was finally achieved by Kutyłowski, by a novel lower bound technique.

THEOREM 1.9 [13]. $\mathrm{CREW}(\mathrm{OR}_n) \geqslant \phi(n)$. *More generally: if* $f \in \mathbf{B}_n$ *and* $|\{\mathbf{a} \in \{0,1\}^n : f(\mathbf{a}) = 1\}| = 2^i \cdot u$, *where* u *is odd, then* $\mathrm{CREW}(f) \geqslant \phi(n-i)$.

In this paper, we will improve upon the results given in [13]. By a new method, we will prove a generalization of Theorem 1.9, which allows us to establish exact lower bounds for many other Boolean functions, including all symmetric functions. Moreover, for arbitrary $f \in \mathbf{B}_n$ the method yields bounds that are optimal up to a constant factor: the lower bound given by the new method is at most a factor 8 smaller than CREW(f). The new approach is based upon the additional structure

given on $\mathbf{B}_n$ by the fact that Boolean functions can be viewed as polynomials with integer coefficients, and the *degree* of such a polynomial can be taken as a complexity measure.

For $S \subseteq \{1, ..., n\}$, we denote the monomial $\prod_{i \in S} x_i$ by m_S. Obviously, each m_S defines a Boolean function. (In this paper we will never consider other than Boolean inputs.) It is well known ([23]; for a short proof see, e.g., [31, p. 6]) that each Boolean function f can be written uniquely as a "polynomial" $\sum_{S \in \mathcal{S}} m_S \pmod 2$, where $\mathcal{S} = \mathcal{S}(f)$, a class of subsets of $\{1, ..., n\}$, is suitably chosen. This representation is called the *ring sum expansion* of f. The number $\max\{|S|: S \in \mathcal{S}(f)\}$ can be regarded as the *degree* of f in this representation. It was used as such in Razborov's lower bound on bounded depth circuits with PARITY gates [22]. The ring sum expansion can be generalized to arbitrary fields F; each $f \in \mathbf{B}_n$ can be written as a linear combination $f = \sum_S \alpha_S(f) \cdot m_S$ for unique coefficients $\alpha_S(f) \in F$. Again, a notion of degree results. Smolensky [27] used this fact for *finite* fields in his approach for proving lower bounds for bounded depth circuits. We will use the field $\mathbf{R}$ of real numbers as the field F; i.e., we use representations of Boolean functions as polynomials with real coefficients and the corresponding notion of degree. This way of representing Boolean functions has been considered on several occasions before, e.g., in connection with work on Fourier transforms of Boolean functions (for a survey see [15]) or on perceptrons [17]. A thorough study of this representation was given by Szegedy in his thesis [29], who was the first to emphasize the role of the degree of the representing polynomial as a complexity measure. Other aspects of this representation are studied in recent work on the Fourier transform of Boolean functions [12, 16] and on polynomial threshold functions [4, 5].

The degree complexity measure is combined with the well-known method of considering a PRAM computation as generating finer and finer (processor and cell) partitions of the input space $\{0, 1\}^n$, as has been used, e.g., in [28, 1, 13]. Two inputs are in the same cell of the partition that belongs to processor P_i at step t if P_i is left in the same state after step t by both inputs. The degrees of the characteristic functions of the classes in these partitions grow as the computation proceeds. We will see that in the case of CREW PRAMs the growth rate can be nicely bounded; hence, a function of large degree has large CREW complexity.

THEOREM 1.10 (Main theorem). $\mathrm{CREW}(f) \geq \phi(\deg(f)) \geq 0.72 \log(\deg(f))$, *for all* $f \in \mathbf{B}_n$.

The main theorem will be proved in Section 3. It can be applied in several ways: Theorem 1.9 can be derived (see Corollary 4.2); in many cases, the degree of a concrete function can be calculated easily; a general exact lower bound for symmetric functions can be proved; finally, whenever the degree of a function f happens to be the same as $D(f)$, its CREW complexity is exactly $\phi(D(f)) + O(1)$ with the additive constant being zero or one. (Note, however, that there are functions f with the property that $\deg(f)$ is substantially smaller than $D(f)$; see Section 2.)

The new technique also yields lower bounds for some variants of the CREW PRAM model. First, we consider a *nondeterministic* model. In each step, each processor may have different alternatives how to proceed, but no matter how the individual processors behave, no write conflicts are allowed to occur. A function f is computed by such a machine if for all inputs $\mathbf{a}$ with $f(\mathbf{a})=1$ there is at least one computation that yields the output 1, and for $\mathbf{a}$ with $f(\mathbf{a})=0$ there is no such computation. For this model, we show the same exact bounds as in the deterministic case, for example, for AND and PARITY. (Note that OR can be computed in constant time in this model—a single processor nondeterministically chooses an input bit, and if this bit equals 1, the processor outputs 1.) Corresponding lower bounds, although weaker by a constant factor, can already be obtained using methods from [18].

Second, we consider a *probabilistic* model. Here, each processor in each step decides by means of a random experiment what computation path to follow. The probability of a write conflict has to be zero. All computations halt after T steps, for T the time bound of the machine. In the unbounded error case (i.e., the machine answers correctly with probability larger than $\frac{1}{2}$) we show the exact lower bound $\phi(n)$ for the PARITY_n function. In the bounded error case (error probability $<\frac{1}{2}(1-\varepsilon)$) our approach combines particularly nicely with a result from [29] that concerns the degree of (real-valued) polynomials that approximate Boolean functions of large critical complexity. We obtain that the (bounded-error) probabilistic CREW complexity of a Boolean function is larger than $\phi(\sqrt{\varepsilon \cdot \text{bc}(f)}) \approx \frac{1}{2}\phi(\text{bc}(f)) - O(\log(1/\varepsilon))$. In combination with Nisan's results this implies that the (bounded-error) probabilistic and the deterministic CREW complexity of f differ at most by a factor of 8.

Finally, we slightly relax the "exclusive-write" restriction and allow that up to k processors write to the same cell simultaneously; the new technique allows us to prove exact lower bounds also for this model. No technique was known before to prove lower bounds for this model that are tight up to a constant factor.

2. Preliminaries

In this section, we review some complexity measures for Boolean functions, in particular the notion of degree; further, a formal description of the abstract CREW PRAM is included.

2.1. *Boolean Functions as Polynomials*

For $S \subseteq \{1, \ldots, n\}$, we let m_S be the (positive) monomial $\prod_{i \in S} x_i$. Clearly, m_S can be regarded as a member of $\mathbf{B}_n$, via $m_S(\mathbf{a}) = \prod_{i \in S} a_i$, for $\mathbf{a} \in \{0,1\}^n$. Obviously, $\mathbf{B}_n$ is a subset of the algebra $F^{\{0,1\}^n}$ for any field F (we shall use $F = \mathbf{R}$). Multiplication of monomials in the algebra $F^{\{0,1\}^n}$ obeys the law $m_S \cdot m_{S'} = m_{S \cup S'}$; that means, monomials multiply just as in the Boolean case.

FACT 2.1 (See [15, 17]). *Every* $f \in \mathbf{R}^{\{0,1\}^n}$ *can be written as a linear combination*

$$f = \sum_{S \subseteq \{1, \ldots, n\}} \alpha_S(f) \cdot m_S$$

with unique coefficients $\alpha_S(f) \in \mathbf{R}$; *if* $f \in \mathbf{B}_n$, *then the* $\alpha_S(f)$ *are integers of absolute value at most* 2^{n-1}.

Proof. For $\mathbf{a} \in \{0,1\}^n$, consider the function $\chi_{\mathbf{a}} \in \mathbf{B}_n$ that takes the value 1 if and only if the input is $\mathbf{a}$. The functions $\chi_{\mathbf{a}}$ form the standard basis of $\mathbf{R}^{\{0,1\}^n}$; more concretely, we have $f = \sum_{\mathbf{a} \in \{0,1\}^n} f(\mathbf{a}) \cdot \chi_{\mathbf{a}}$, for arbitrary $f \in \mathbf{R}^{\{0,1\}^n}$. It is obvious that

$$\chi_{\mathbf{a}} = \Bigg(\prod_{\substack{1 \leqslant i \leqslant n \\ a_i = 1}} x_i \Bigg) \cdot \Bigg(\prod_{\substack{1 \leqslant i \leqslant n \\ a_i = 0}} (1 - x_i) \Bigg),$$

whence (by expanding the product) it follows that $\chi_{\mathbf{a}} = \sum_{S \subseteq \{1, \ldots, n\}} \alpha_S(\chi_{\mathbf{a}}) \cdot m_S$, where

$$\alpha_S(\chi_{\mathbf{a}}) = \begin{cases} (-1)^{|\{i \in S : a_i = 0\}|}, & \text{if } \{i : a_i = 1\} \subseteq S, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Let $\|\mathbf{a}\|$ denote the number of 1's in $\mathbf{a}$ (the *weight* of $\mathbf{a}$). It is clear that for $\mathbf{a}$ with $\{i : a_i = 1\} \subseteq S$ we have

$$|\{i \in S : a_i = 0\}| \equiv |S| - \|\mathbf{a}\| \equiv |S| + \mathrm{PARITY}_n(\mathbf{a}) \pmod 2.$$

Thus it follows from (1) that an arbitrary function $f \in \mathbf{R}^{\{0,1\}^n}$ can be written as $f = \sum_{S \subseteq \{1, \ldots, n\}} \alpha_S(f) \cdot m_S$, where the coefficients $\alpha_S(f)$ are given by the equation

$$\alpha_S(f) = \sum_{\mathbf{a} \in \{0,1\}^n} f(\mathbf{a}) \cdot \alpha_S(\chi_{\mathbf{a}}) = (-1)^{|S|} \cdot \sum_{\substack{\mathbf{a} \in \{0,1\}^n \\ \{i : a_i = 1\} \subseteq S}} f(\mathbf{a}) \cdot (-1)^{\mathrm{PARITY}_n(\mathbf{a})}. \tag{2}$$

We have thus shown that the 2^n functions m_S, $S \subseteq \{1, \ldots, n\}$, span the vector space $\mathbf{R}^{\{0,1\}^n}$. Since this space has dimension 2^n, it follows that the functions m_S form a basis, and hence that the coefficients $\alpha_S(f)$ are uniquely determined.

Finally, it can be read from Eq. (2) that if f is a Boolean function, then $\alpha_S(f)$ is an integer; moreover, since the number of 1's (and (-1)'s, respectively) in the last sum in (2) does not exceed 2^{n-1}, we also have $|\alpha_S(f)| \leqslant 2^{n-1}$. ∎

Fact 2.1 justifies the following definition.

DEFINITION 2.2 [27]. $\deg(f) := \max\{|S| : \alpha_S(f) \neq 0\}$, for $f \in \mathbf{R}^{\{0,1\}^n}$.

We note some laws concerning the degree, which are easily verified.

LEMMA 2.3. *For arbitrary* $f, g \in \mathbf{B}_n$ *the following assertions hold*:

(a) $\deg(f \wedge g) = \deg(f \cdot g) \leqslant \deg(f) + \deg(g)$.

(b) $\deg(\bar{f}) = \deg(1 - f) = \deg(f)$.

(c) $\deg(f \vee g) = \deg(1 - (1-f)(1-g)) = \deg(f + g - f \cdot g) \leqslant \deg(f) + \deg(g)$.

(d) *If* $f \wedge g \equiv 0$, *then* $\deg(f \vee g) = \deg(f + g) \leqslant \max\{\deg(f), \deg(g)\}$.

(e) $\deg(f \wedge \bar{g}) = \deg(f \cdot (1-g)) \leqslant \deg(f) + \deg(g)$.

(f) *The degree is invariant under negation of variables, e.g.,* $\deg(f(\bar{x}_1, x_2, \ldots, x_n)) = \deg(f)$. *Thus,* $\deg(f^{\text{dual}}) = \deg(\overline{f(\bar{x}_1, \ldots, \bar{x}_n)}) = \deg(f)$.

(g) *If* g *is a subfunction of* f, *i.e., it results from* f *by fixing some input values to* 0 *or* 1, *then* $\deg(g) \leqslant \deg(f)$.

Further, we list some facts concerning the relationship between degree and other complexity measures for Boolean functions, which were defined in the Introduction.

FACT 2.4. $\deg(f) \leqslant D(f)$.

Proof. Straightforward. ∎

FACT 2.5 [29]. $\deg(f) \geqslant \sqrt{\mathrm{bc}(f)}$. (*Hence,* $\deg(f) \geqslant \sqrt{\mathrm{c}(f)}$.)

This is a special case of the following.

FACT 2.6 ([29], variant of Theorem 2.2.3). *If* $f \in \mathbf{B}_n$, *and* $h \in \mathbf{R}^{\{0,1\}^n}$ *satisfies* $|h(\mathbf{a}) - f(\mathbf{a})| \leqslant \frac{1}{2}(1-\varepsilon)$ *and* $0 \leqslant h(\mathbf{a}) \leqslant 1$ *for all* $\mathbf{a} \in \{0,1\}^n$, *then* $\deg(h) \geqslant \sqrt{\varepsilon \cdot \mathrm{bc}(f)}$.

FACT 2.7 [18]. $\mathrm{bc}(f) \geqslant \sqrt{\mathrm{s}(f)}$.

FACT 2.8 [3, 11, 30]. $\mathrm{s}(f) \geqslant \sqrt{D(f)}$.

COROLLARY 2.9. $\deg(f) \geqslant D(f)^{1/8}$.

Proof. Combine Facts 2.5, 2.7, and 2.8. ∎

It is worth noting that there are functions $f \in \mathbf{B}_n$ for which $\deg(f)$ is substantially smaller than $D(f)$ and also functions for which the main theorem (Theorem 1.10) does not give a tight lower bound.

EXAMPLE 2.10. In [20] a function f is constructed (by giving a PRAM program) with $\mathrm{c}(f) = n$, hence $D(f) = n$, and $\mathrm{CREW}(f) \leqslant \log_{(2+\sqrt{2})} n$; hence (by the main theorem) $\log_b(\deg(f)) \leqslant \log_{(2+\sqrt{2})} n$, for $b = [\frac{1}{2}(1+\sqrt{5})]^2$. This implies that $\deg(f) \leqslant n^{\log(b)/\log(2+\sqrt{2})} < n^{0.79}$.

EXAMPLE 2.11 [2, 19]. Define

$$g_1(a_1, a_2, a_3) := \begin{cases} 1, & \text{if either 1 or 2 inputs are 1,} \\ 0, & \text{otherwise.} \end{cases}$$

Then define recursively, for $k \geqslant 2$, $n = 3^k$,

$$g_k(a_1, ..., a_n) := g_1(g_{k-1}(a_1, ..., a_{n/3}), g_{k-1}(a_{n/3+1}, ..., a_{2n/3}), g_{k-1}(a_{2n/3+1}, ..., a_n)).$$

The degree of g_1 is 2, so, by induction, the degree of g_k is $2^k = n^{1/\log 3} \approx n^{0.63}$. On the other hand, the input $(0, ..., 0)$ is critical, i.e., changing any input bit changes the value of g_k, whence $D(g_k) \geqslant c(g_k) = n = 3^k$. By Theorem 1.6 we obtain $\mathrm{CREW}(g_k) \geqslant \frac{1}{2} \log 3^k \approx 0.79k$. However, $\phi(\deg(g_k)) \leqslant 0.73k + 1.34$; so the lower bound given by the main theorem is not tight.

2.2. *CREW PRAMs and Partitions*

In order to have a sound basis for the lower bound proof, we recall the formal definition of the abstract CREW PRAM from [7].

DEFINITION 2.12. The components of a CREW PRAM are the following: the processors $P_1, P_2, ...$; the memory cells $C_1, C_2, ...$; an alphabet Σ with $\{0, 1, B\} \subseteq \Sigma$; a set Q of states; a number n of input bits; a running time T; an output function $out: \Sigma \to \{0, 1\}$. Associated with each processor P_i there are a distinguished initial state $q_i^0 \in Q$; a read-address function $\rho_i: Q \to \mathbf{N}$; a state transition function $\delta_i: Q \times \Sigma \to Q$; a write-value function $\sigma_i: Q \to \Sigma$; and a write-address function $\tau_i: Q \to \mathbf{N}$. (These functions constitute the program of the PRAM.) For each given input $\mathbf{a} = (a_1, ..., a_n) \in \{0, 1\}^n$ the computation proceeds in steps $t = 0, 1, 2, ..., T$. At the end of each step each processor P_i is in some state $q_i^t \in Q$, and each cell C_j contains some symbol $s_j^t \in \Sigma$. Initially (i.e., after step 0) processor P_i is in state q_i^0, for $i \in \mathbf{N}$, and cell C_j contains either the jth input bit a_j (if $1 \leqslant j \leqslant n$) or B (if $j > n$). Step $t \in \{1, ..., T\}$ consists of two phases performed simultaneously by all processors:

(i) *Reading and internal computation.* P_i determines $j := \rho_i(q_i^{t-1})$, reads the symbol $s = s_j^{t-1} \in \Sigma$ contained in C_j after step $t-1$, and switches state to $q_i^t = \delta_i(q_i^{t-1}, s)$. (We may assume w.l.o.g. that P_i reads in every step.)

(ii) *Writing.* P_i determines $s' := \sigma_i(q_i^t)$ and $j' := \tau_i(q_i^t)$ and writes the symbol s' to cell $C_{j'}$. (If $j' = 0$, then P_i does not write in step t.) For any $j \geqslant 1$, the content of C_j after step t is either s_j^{t-1} (if $\tau_i(q_i^t) \neq j$ for all i) or $\sigma_i(q_i^t)$ for the unique processor P_i with $\tau_i(q_i^t) = j$. It must not happen that in any one step two different processors write to the same cell.

The result of the computation is the value $\mathrm{out}(s_1^T)$, where s_1^T is the symbol contained in C_1 after step T. The PRAM computes a Boolean function f if the result of the computation on input $\mathbf{a}$ is $f(\mathbf{a})$, for all $\mathbf{a} \in \{0, 1\}^n$.

In the following, we always think of a CREW PRAM M as being given and drop the subscript M from the notation. We will assume, without loss of generality, that M is a "full information" machine: the processors remember everything they have read so far and when a processor writes, then the symbol it writes codes all the information it has collected so far, as well as the number of the processor and the current step number. The output function translates the "full information symbol" written in C_1 after step T to the output bit.

It has been noted several times before (e.g., [28, 1, 13]) that a useful way of looking at the computations of PRAMs is to study the partitions of $\{0,1\}^n$ induced by the states and cell contents during the computations.

DEFINITION 2.13. Let a CREW PRAM be given. For $0 \leqslant t \leqslant T$, $i, j \in \mathbf{N}$, a state $q \in Q$, and a symbol $s \in \Sigma$ define

$$G(q, i, t) := \{\mathbf{a} \in \{0,1\}^n : \text{processor } P_i \text{ is in state } q \text{ after step } t \text{ in the computation on input } \mathbf{a}\},$$

$$H(s, j, t) := \{\mathbf{a} \in \{0,1\}^n : \text{cell } C_j \text{ contains } s \text{ after step } t \text{ in the computation on input } \mathbf{a}\}.$$

Further let, for $0 \leqslant t \leqslant T$,

$$\mathscr{G}(t) := \{G(q, i, t) : i \in \mathbf{N}, q \in Q\},$$

$$\mathscr{H}(t) := \{H(s, j, t) : j \in \mathbf{N}, s \in \Sigma\}.$$

Clearly, $\{G(q, i, t) : q \in Q\}$ and $\{H(s, j, t) : s \in \Sigma\}$ are partitions of $\{0,1\}^n$ for each (i, t) and (j, t). Informally, for any input $\mathbf{a}$ in $G(q, i, t)$, P_i after t steps only knows that the input is in the set $G(q, i, t)$; similarly for the "information" described by the fact that cell C_j contains symbol s after step t. Already in [1] it was noted that the classes of these partitions are essentially the states; in [13] it was demonstrated that a variant of the CREW PRAM (the "Boolean PRAM") can be defined, which is equivalent to the original model and in which the classes of these partitions are used explicitly as states of processors and cell contents. Although the Boolean PRAM would be a convenient framework for our lower bound proof, too, we stick to the standard definition, since this is more convenient for the extension to "few-write" PRAMs in Section 7.

3. PROOF OF THE MAIN THEOREM

THEOREM 3.1. $\mathrm{CREW}(f) \geqslant \phi(\deg(f))$, *for all* $f \in \mathbf{B}_n$.

Notation. For $A \subseteq \{0,1\}^n$, we denote the characteristic function of A by χ_A or $\chi(A)$. For $\mathscr{A}$ a class of subsets of $\{0,1\}^n$, let $\deg(\mathscr{A}) := \max\{\deg(\chi_A) : A \in \mathscr{A}\}$.

Proof of Theorem 3.1. Since $f^{-1}(1)$ is the disjoint union of all sets $H(s, 1, T)$ with $\text{out}(s) = 1$, it suffices to show, in view of Lemma 2.3(d), that $\deg(\mathscr{H}(T)) \leqslant F_{2T+1}$. This is immediate from assertion (d) in the following main lemma.

LEMMA 3.2. (a) $\deg(\mathscr{G}(0)) = 0$ *and* $\deg(\mathscr{H}(0)) = 1$.

(b) $\deg(\mathscr{G}(t)) \leqslant \deg(\mathscr{H}(t-1)) + \deg(\mathscr{G}(t-1))$, *for* $0 < t \leqslant T$.

(c) $\deg(\mathscr{H}(t)) \leqslant \deg(\mathscr{H}(t-1)) + \deg(\mathscr{G}(t))$, *for* $0 < t \leqslant T$.

(d) $\deg(\mathscr{G}(t)) \leqslant F_{2t}$ *and* $\deg(\mathscr{H}(t)) \leqslant F_{2t+1}$, *for* $0 \leqslant t \leqslant T$.

Proof. First note that (d) follows by induction from (a)–(c) and the definition of the Fibonacci numbers. We prove (a)–(c).

(a) It is immediate from the initialization conditions that $\mathscr{G}(0) = \{\varnothing, \{0, 1\}^n\}$ (the initial state of P_i does not depend on the input). The characteristic functions of these two sets are represented by the constant polynomials 0 and 1 of degree 0. If $H \in \mathscr{H}(0)$, then χ_H is one of the functions x_j and $1 - x_j$ (for $1 \leqslant j \leqslant n$, the cell C_j contains the jth input bit), or 0 and 1 (for $j > n$, the content of C_j is independent of the input). These functions have degree 1 and 0, respectively.

(b) Let $G' = G(q', i, t) \neq \varnothing$ for some $q' \in Q$, $i \geqslant 1$, and let $\mathbf{a} \in G'$. In the computation on $\mathbf{a}$, processor P_i was in some state q at the end of step $t-1$, and read some cell C_j in step t, where $j = \rho_i(q)$. It is easily seen that $G' = G \cap H$ for $H = H(s, j, t-1)$ and $G = G(q, i, t-1)$, where s is such that $\mathbf{a} \in H(s, j, t-1)$. (For the inclusion "$\subseteq$" we use the "full information" assumption: the previous state q and the symbol s read can be recovered from q'.) By Lemma 2.3(a), we obtain $\deg(\chi_{G'}) = \deg(\chi_{G \cap H}) = \deg(\chi_G \cdot \chi_H) \leqslant \deg(\chi_G) + \deg(\chi_H)$.

(c) Let $H' = H(s, j, t) \neq \varnothing$ for some $s \in \Sigma$, $j \geqslant 1$, and let $\mathbf{a} \in H'$. We consider two cases.

Case 1. *On input* $\mathbf{a}$, *some processor* P_i *writes to* C_j *in step* t. Let q be the state of P_i after step t. Clearly, for all inputs in $G(q, i, t)$ processor P_i will write the same symbol s to C_j in step t, since $\tau_i(q) = j$ and $\sigma_i(q) = s$. By the "full information" assumption, for inputs not in $G(q, i, t)$ the symbol contained in C_j after step t will be different from s. Hence, $H' = G(q, i, t) \in \mathscr{G}(t)$, thus $\deg(\chi_{H'}) \leqslant \deg(\mathscr{G}(t))$.

Case 2. *On input* $\mathbf{a}$, *no processor* P_i *writes to* C_j *in step* t. Let $G_1 = G(q_1, i_1, t), \ldots, G_r = G(q_r, i_r, t)$ be a list of all (nonempty) classes of processor partitions in step t so that P_{i_u} writes to C_j in step t; that means $\tau_{i_u}(q_{i_u}) = j$, for $1 \leqslant u \leqslant r$. First note that $G_1, \ldots, G_r$ are disjoint: if $\mathbf{b}$ were an element of $G_u \cap G_v$, $u \neq v$, $i_u \neq i_v$, then on input $\mathbf{b}$ both P_{i_u} and P_{i_v} would write to C_j, contradicting the exclusive-write rule; if $u \neq v$ and $i_u = i_v$, then the classes G_u and G_v are obviously disjoint. By the "full information" assumption, the symbol written by P_{i_u} in state q_{i_u} is different from s, for $1 \leqslant u \leqslant r$; hence we have $H' = H - G$ for $H := H(s, j, t-1)$ and $G := G_1 \cup \cdots \cup G_r$. Since $G_1, \ldots, G_r$ are disjoint, Lemma 2.3(d) implies $\deg(\chi_G) \leqslant \deg(\mathscr{G}(t))$. Further, we obtain by Lemma 2.3(e) that $\deg(\chi_{H'}) \leqslant \deg(\chi_H) + \deg(\chi_G)$. ∎

4. Applications and Examples

4.1. *Consequences of the Main Theorem*

First, we obtain an alternative proof of the classic lower bound from [7].

Corollary 4.1. $\mathrm{CREW}(f) \geq \phi(\sqrt{\mathrm{c}(f)}) \geq 0.36 \log(\mathrm{c}(f))$, *for all* $f \in \mathbf{B}_n$.

Proof. Combine Theorem 3.1 with Fact 2.5. ▮

Next, we note that the main result of [13] can be derived directly.

Corollary 4.2 [13]. $\mathrm{CREW}(\mathrm{OR}_n) \geq \phi(n)$. *More generally, if* $f \in \mathbf{B}_n$ *and* $|f^{-1}(1)| = 2^i \cdot u$ *for* u *odd, then* $\mathrm{CREW}(f) \geq \phi(n-i)$.

Proof. Let $r := \deg(f)$. Then $f = \sum_{|S| \leq r} \alpha_S(f) \cdot m_S$. Hence

$$|f^{-1}(1)| = \sum_{\mathbf{a}} f(\mathbf{a}) = \sum_{\mathbf{a}} \sum_{|S| \leq r} \alpha_S(f) \cdot m_S(\mathbf{a})$$

$$= \sum_{|S| \leq r} \alpha_S(f) \cdot \sum_{\mathbf{a}} m_S(\mathbf{a}) = \sum_{|S| \leq r} \alpha_S(f) \cdot |m_S^{-1}(1)|.$$

Since $\alpha_S(f)$ is an integer (by Fact 2.1) and $|m_S^{-1}(1)| = 2^{n-|S|}$ is divisible by 2^{n-r} for $|S| \leq r$, this implies that 2^{n-r} divides $|f^{-1}(1)|$; hence $n - r \leq i$. By Theorem 3.1, $\mathrm{CREW}(f) \geq \phi(r) \geq \phi(n-i)$. ▮

Corollary 4.3. $\mathrm{CREW}(\mathrm{PARITY}_n) \geq \phi(n)$. *More generally, if* $f \in \mathbf{B}_n$ *and*

$$|\{\mathbf{a} : f(\mathbf{a}) = 1 \wedge \mathrm{PARITY}_n(\mathbf{a}) = 0\}| \neq |\{\mathbf{a} : f(\mathbf{a}) = 1 \wedge \mathrm{PARITY}_n(\mathbf{a}) = 1\}|,$$

then $\mathrm{CREW}(f) \geq \phi(n)$.

Proof. From formula (2) in the proof of Fact 2.1 we obtain

$$|\alpha_{\{1, \ldots, n\}}(f)| = \left| \sum_{\mathbf{a} \in f^{-1}(1)} (-1)^{\mathrm{PARITY}_n(\mathbf{a})} \right|.$$

Thus, $\deg(f) < n$ if and only if $f^{-1}(1)$ contains the same number of elements with even and with odd parity. Now apply Theorem 3.1. ▮

Remark. A more involved analysis shows that $\mathrm{CREW}(\mathrm{PARITY}_n) \geq \min\{t : F_{2t} \geq n\}$, which is often one step more than $\phi(n)$ [14].

Next we consider "nondegenerate" Boolean functions. Assume that $f \in \mathbf{B}_n$ depends on all n variables, i.e., for all $i \in \{1, \ldots, n\}$ there is an $\mathbf{a} \in \{0,1\}^n$ with $f(\mathbf{a}) \neq f(\mathbf{a}^{\{i\}})$. The main result of [26] says that $\mathrm{c}(f) \geq (\frac{1}{2} - o(1)) \log n$. In combination with Theorem 1.6 this entails that $\mathrm{CREW}(f) \geq \frac{1}{2} \log \log n - O(1)$. We obtain a slightly larger bound by combining a theorem from [19] with Theorem 3.1.

COROLLARY 4.4. *If $f \in \mathbf{B}_n$ depends on all n variables, then*

$$\mathrm{CREW}(f) \geqslant \phi(\log n) - O(1) \geqslant 0.72 \log\log n - O(1).$$

Proof. In [19] it is shown that such f satisfy $\deg(f) \geqslant \log n - O(\log\log n) \geqslant \frac{1}{2}\log n$. Now apply Theorem 3.1. ∎

The last corollary concerns the CREW complexity of almost all Boolean functions. In [6] it is shown that $\mathrm{c}(f) \geqslant n-1$ for almost all $f \in \mathbf{B}_n$, that is,

$$\lim_{n\to\infty} \frac{|\{f \in \mathbf{B}_n : \mathrm{c}(f) \geqslant n-1\}|}{2^{2^n}} = 1.$$

By Theorem 1.6 it follows that almost all Boolean functions satisfy $\mathrm{CREW}(f) \geqslant \frac{1}{2}\log n - o(1)$. Utilizing Theorem 3.1, we may determine the CREW complexity of almost all Boolean functions up to a small additive constant. (Recall that, according to Theorem 1.4, $\mathrm{CREW}(f) \leqslant \phi(n) + 1$ for all $f \in \mathbf{B}_n$.)

COROLLARY 4.5. *Almost all Boolean functions $f \in \mathbf{B}_n$ satisfy* $\mathrm{CREW}(f) \geqslant \phi(n)$; *that is,*

$$\lim_{n\to\infty} \frac{|\{f \in \mathbf{B}_n : \mathrm{CREW}(f) \geqslant \phi(n)\}|}{2^{2^n}} = 1.$$

Proof. In [24, Corollary 3.4], it is shown that almost all functions $f \in \mathbf{B}_n$ (namely, at least a fraction of about $1 - n^{-1/2}$) satisfy

$$|\{\mathbf{a} : f(\mathbf{a}) = 1 \wedge \mathrm{PARITY}_n(\mathbf{a}) = 0\}| \neq |\{\mathbf{a} : f(\mathbf{a}) = 1 \wedge \mathrm{PARITY}_n(\mathbf{a}) = 1\}|.$$

The claim now follows by applying Corollary 4.3. ∎

4.2. *Monotone Functions*

For $\mathbf{a}, \mathbf{b} \in \{0,1\}^n$ we write $\mathbf{a} \leqslant \mathbf{b}$ if $\forall_{i \in \{1,\ldots,n\}}\, a_i \leqslant b_i$, and $\mathbf{a} < \mathbf{b}$ if $\mathbf{a} \leqslant \mathbf{b}$ and $\mathbf{a} \neq \mathbf{b}$. In this subsection, let $f \in \mathbf{B}_n$ be monotone, that is, $f(\mathbf{a}) \leqslant f(\mathbf{b})$ whenever $\mathbf{a} \leqslant \mathbf{b}$. Let

$$\mathrm{MIN}(f) := \{\mathbf{a} \in \{0,1\}^n : f(\mathbf{a}) = 1 \text{ and } \forall_{\mathbf{b} < \mathbf{a}} f(\mathbf{b}) = 0\},$$
$$\mathrm{MAX}(f) := \{\mathbf{a} \in \{0,1\}^n : f(\mathbf{a}) = 0 \text{ and } \forall_{\mathbf{b} > \mathbf{a}} f(\mathbf{b}) = 1\}.$$

$\mathrm{MIN}(f)$ ($\mathrm{MAX}(f)$) is the set of minimal (maximal) elements in $f^{-1}(1)$ ($f^{-1}(0)$) with respect to the partial order $\leqslant$ on $\{0,1\}^n$. It is well known (see, e.g., [6 or 31]) that

$$\mathrm{s}(f) = \mathrm{c}(f) = \max\{k : k = \|\mathbf{a}\| \text{ for some } \mathbf{a} \in \mathrm{MIN}(f) \text{ or } k = n - \|\mathbf{a}\| \text{ for some } \mathbf{a} \in \mathrm{MAX}(f)\},$$

where $\|\mathbf{a}\|$ denotes the number of 1's in the vector $\mathbf{a}$ (the weight of $\mathbf{a}$). Concerning the degree of monotone functions, we have the following.

LEMMA 4.6. *If $f \in \mathbf{B}_n$ is monotone, then* $\deg(f) \geq c(f)$.

Proof. We show that $\deg(f) \geq \|\mathbf{a}\|$ for all $\mathbf{a} \in \mathrm{MIN}(f)$. Let $\mathbf{a} \in \mathrm{MIN}(f)$ be arbitrary. Then the subfunction of f obtained by fixing all variables x_i with $a_i = 0$ to 0 is the AND of $\|\mathbf{a}\|$ variables, which has degree $\|\mathbf{a}\|$. By Lemma 2.3(g) we obtain $\deg(f) \geq \|\mathbf{a}\|$. Similarly, we obtain $\deg(f) \geq n - \|\mathbf{a}\|$ for all $\mathbf{a} \in \mathrm{MAX}(f)$. ∎

We may now determine $\mathrm{CREW}(f)$ for an arbitrary monotone $f \in \mathbf{B}_n$ up to a factor of 2.

COROLLARY 4.7. *If $f \in \mathbf{B}_n$ is monotone, then*

(a) $\mathrm{CREW}(f) \geq \phi(c(f)) \geq 0.72 \log(c(f))$.

(b) $1.45 \log(c(f)) + 1 \geq \phi(c(f)^2) + 1 \geq \mathrm{CREW}(f)$.

Proof. Part (a) follows directly from Theorem 3.1 and Lemma 4.6. Part (b) follows from $D(f) \leq s(f)^2$ (Fact 2.8), $\mathrm{CREW}(f) \leq \phi(D(f)) + 1$ (Theorem 1.4(c)), and $c(f) = s(f)$ (since f is monotone). ∎

In some cases, the degree of a monotone function f is substantially bigger than $c(f)$, thus we obtain a better bound by using Theorem 3.1 directly.

EXAMPLE 4.8. Let $n = k^2$, and let

$$f = (x_1 \wedge \cdots \wedge x_k) \vee (x_{k+1} \wedge \cdots \wedge x_{2k}) \vee \cdots \vee (x_{n-k+1} \wedge \cdots \wedge x_n).$$

It is easily checked that $c(f) = s(f) = k$. On the other hand, $f = 1 - \prod_{j=1}^{k} (1 - x_{(j-1)k+1} \cdots x_{jk})$ obviously satisfies $\alpha_{\{1, \ldots, n\}}(f) = (-1)^k$; hence $\deg(f) = n$. The resulting bound $\mathrm{CREW}(f) \geq \phi(n)$ is by a factor of 2 bigger than the lower bound $\phi(k)$ given by Corollary 4.7. Incidentally, note that Fact 2.4 implies $D(f) = n$; thus we have a simple example of a monotone function with a quadratic gap between $s(f)$ and $D(f)$. (This is as large as it can become, see Fact 2.8.)

4.3. *Symmetric Functions*

In this subsection, we consider symmetric functions, i.e., functions $f \in \mathbf{B}_n$ that satisfy $f(a_1, \ldots, a_n) = f(a_{\pi(1)}, \ldots, a_{\pi(n)})$ for all permutations π of $\{1, \ldots, n\}$, and all $\mathbf{a} \in \{0, 1\}^n$. It is well known (see, e.g., [6]) that $c(f) \geq \lceil \frac{1}{2}(n+1) \rceil$ for all nonconstant symmetric functions $f \in \mathbf{B}_n$; thus, by Theorem 1.6 we have $\mathrm{CREW}(f) \geq \frac{1}{2}(\log n - 1)$ for all such f. In the following, we derive *exact* lower bounds on the CREW complexity of all nonconstant symmetric functions.

For arbitrary symmetric $f \in \mathbf{B}_n$, it is easy to see that $\alpha_S(f)$ only depends on the size of S. More precisely, if $U(f)$ denotes the set of all k so that $f(\mathbf{a}) = 1$ if $\mathbf{a}$ contains k 1's, then it follows from (2) in the proof of Fact 2.1 that $\alpha_S(f) = (-1)^{|S|} \cdot \sum_{k \in U(f)} (-1)^k \cdot \binom{|S|}{k}$. For many symmetric functions, among them the exactly-k-functions and the threshold functions (cf. [31]), this formula already

implies $\deg(f)=n$. There are nontrivial symmetric functions with degree $n-1$. However, note the following.

THEOREM 4.9. *If $f\in \mathbf{B}_n$ is symmetric and nonconstant, then* $\deg(f)>\frac{1}{2}n$; *hence* $\mathrm{CREW}(f)\geqslant \phi(n)-1$.

Proof. By contradiction. Suppose that $r:=\deg(f)\leqslant \frac{1}{2}n$. Since f is not constant, $r>0$. Choose $S\subseteq\{1,\ldots,n\}$, $|S|=2r$. Since f is Boolean, $f\cdot f=f$. For $\alpha_S(f)$ this yields

$$\alpha_S(f)=\sum_{\substack{S',\,S''\subseteq S\\ S'\cup S''=S}}\alpha_{S'}(f)\,\alpha_{S''}(f)=\sum_{\substack{S'\subseteq S\\ |S'|=r}}\alpha_{S'}(f)\,\alpha_{S-S'}(f).$$

(The second equality holds since $\alpha_Z(f)=0$, if $|Z|>r$.) By symmetry, there is some number $\alpha\neq 0$ such that $\alpha_{S'}(f)=\alpha_{S-S'}(f)=\alpha$ for all $S'\subseteq S$ with $|S'|=r$. Hence each summand in the last sum equals α^2, and thus $\alpha_S(f)=\binom{2r}{r}\cdot\alpha^2\neq 0$, a contradiction. ∎

5. NONDETERMINISTIC CREW PRAMS

In this section, we extend the previous results to *nondeterministic* CREW PRAMs (NCREW PRAMs). By this, we mean the following variant of the CREW PRAM model. In each step, each processor may have several alternatives how to proceed. Technically, the transition functions δ_i are replaced by multiple-valued functions $\delta_i: Q\times\Sigma\to 2^Q$. In step t, if processor P_i is in state q_i^{t-1} and reads symbol s, it enters *some* state q_i^t from $\delta_i(q_i^{t-1},s)$. As usual, we say that $f\in\mathbf{B}_n$ is computed by an NCREW PRAM in T steps if for all inputs $\mathbf{a}$ the following holds: $f(\mathbf{a})=1$ if and only if some computation path leads to a symbol $s\in\mathrm{out}^{-1}(1)$ being contained in cell C_1 after step T. We require that no matter which one of the possible transitions the individual processors choose in the course of a computation, no write conflict will occur. Let $\mathrm{NCREW}(f)$ denote the minimal T so that an NCREW PRAM computes f in T steps. We write $g\leqslant f$ if $g(\mathbf{a})\leqslant f(\mathbf{a})$ for all inputs $\mathbf{a}\in\{0,1\}^n$.

LEMMA 5.1. (a) *If $f\in\mathbf{B}_n$ satisfies* $\mathrm{NCREW}(f)\leqslant T$, *then* $f=\bigvee_{g\in\Gamma} g$ *for a set Γ of Boolean functions with* $\mathrm{CREW}(g)\leqslant T$ *for all $g\in\Gamma$.*

(b) *If $f\in\mathbf{B}_n$ can be written in the way described in* (a), *then* $\mathrm{NCREW}(f)\leqslant T+2$.

Proof. (a) Fix an NCREW PRAM that computes f in T steps. Consider an arbitrary input $\mathbf{a}\in f^{-1}(1)$. Choose an "accepting" computation of the PRAM on input $\mathbf{a}$, i.e., a computation that leaves some $s\in\mathrm{out}^{-1}(1)$ in cell C_1. Fix the decisions of the processors in such a way that this computation is possible. The resulting deterministic CREW PRAM makes T steps and computes some $g_{\mathbf{a}}\in\mathbf{B}_n$ with $g_{\mathbf{a}}(\mathbf{a})=1$ and $g_{\mathbf{a}}\leqslant f$. Let $\Gamma:=\{g_{\mathbf{a}}: f(\mathbf{a})=1\}$. Then $\bigvee_{g\in\Gamma} g\leqslant f$, since $g\leqslant f$ for each $g\in\Gamma$. If $\bigvee_{g\in\Gamma} g$ were different from f, there would be an $\mathbf{a}$ such that $f(\mathbf{a})=1$ and $\bigvee_{g\in\Gamma} g(\mathbf{a})=0$. But this is impossible, since $g_{\mathbf{a}}(\mathbf{a})=1$ and $g_{\mathbf{a}}\in\Gamma$.

(b) In steps 1 and 2, processor P_1 nondeterministically chooses $g \in \Gamma$ and communicates (a T-step CREW PRAM program for) g to all other processors. In step 3 through $T+2$, the processors deterministically compute g on the actual input. ▮

THEOREM 5.2. (a) *If* $0 \neq f \leqslant \text{PARITY}_n$, *then* $\text{NCREW}(f) \geqslant \phi(n)$.

(b) $\text{NCREW}(f) \geqslant \phi(n - \lfloor \log |f^{-1}(1)| \rfloor)$, *for all* $f \in \mathbf{B}_n$, $f \neq 0$.

Proof. (a) Apply Lemma 5.1 and Corollary 4.3.

(b) Let $T = \text{NCREW}(f)$. By Lemma 5.1, there is some nonconstant g with $g^{-1}(1) \subseteq f^{-1}(1)$ so that $\text{CREW}(g) \leqslant T$. By Corollary 4.2, $|g^{-1}(1)|$ is divisible by $2^{n-F_{2T+1}}$; hence $|f^{-1}(1)| \geqslant 2^{n-F_{2T+1}}$. The result follows. ▮

COROLLARY 5.3. *Computing* PARITY_n, $1 - \text{PARITY}_n$, *or* AND_n *takes at least* $\phi(n)$ *steps on NCREW PRAMs.* (*Note that* OR_n *can be computed in constant time on an NCREW PRAM.*)

Remark. We briefly discuss the relationship of $\text{NCREW}(f)$ to other complexity measures. These considerations are based on Lemma 5.1 and the results from [18]. First, we consider a variant of the definition of the sensitive complexity $s(f)$ (cf. Definition 1.7(b)). For $\eta \in \{0, 1\}$, we denote by $s_\eta(f)$ the maximum of the numbers $\min\{k: \exists_{S \subseteq \{1, \ldots, n\},\, |S| = k} f(\mathbf{a}) = f(\mathbf{b}) \text{ if } \forall_{i \in S}\, a_i = b_i\}$ taken over all inputs $\mathbf{a}$ with $f(\mathbf{a}) = \eta$. Now assume $\text{NCREW}(f) = T$. By Lemma 5.1, $f = \bigvee_{g \in \Gamma} g$ for a set $\Gamma \subseteq \mathbf{B}_n$ with $\text{CREW}(g) \leqslant T$ for $g \in \Gamma$. By the precise version of Theorem 1.8 (which makes the constants explicit) it follows that the decision tree complexity $D(g)$ is bounded by 2^{8T} for all $g \in \Gamma$. It easily follows from the definitions that $s_1(f) \leqslant \max_{g \in \Gamma} D(g)$; thus $s_1(f) \leqslant 2^{8T}$; i.e., $\text{NCREW}(f) \geqslant \frac{1}{8} \log(s_1(f))$. On the other hand, it follows from Lemma 5.1(b) and Theorem 1.4 that $\text{NCREW}(f) \leqslant 0.7203 \log(s_1(f)) + 2.34$. Summing up,

$$\tfrac{1}{8} \log(s_1(f)) \leqslant \text{NCREW}(f) \leqslant 0.7203 \log(s_1(f)) + 2.34.$$

This relationship can be utilized to prove lower bounds for $\text{NCREW}(f)$ by determining $s_1(f)$. In contrast, Theorem 5.2 gives *exact* lower bounds in many cases.

A second observation, which links deterministic and nondeterministic complexities on CREW PRAMs, may also be of interest. Since, by a more precise version of Fact 2.8, $D(f) \leqslant s_0(f) \cdot s_1(f)$, and since $s_0(f) = s_1(\bar{f})$, we obtain the following from Theorem 1.4(c) and the inequalities in the previous paragraph:

$$\begin{aligned} \text{CREW}(f) &\leqslant 0.7203 \log(D(f)) + 2.34 \\ &\leqslant 0.7203(\log(s_0(f)) + \log(s_1(f))) + 2.34 \\ &\leqslant 5.8(\text{NCREW}(f) + \text{NCREW}(\bar{f})) + 2.34. \end{aligned}$$

6. Probabilistic CREW PRAMs

In this section, applications of the lower bound method of Section 3 to *probabilistic* PRAMs are discussed. A probabilistic CREW PRAM (abbreviated PCREW PRAM) is obtained from the standard model (cf. Section 2) as follows. In each step t, upon reading a symbol, each processor P_i chooses one of several possible new states according to some probability distribution on Q, the set of all states. This distribution only depends on the old state P_i was in and on the symbol s read by P_i in step t. The random choices of the different processors and those of the same processor at different steps are assumed to be independent. We demand that, for each fixed input, the probability of two processors writing to the same cell in the same step is zero. The result of a computation is out(s), where $s \in \Sigma$ is the content of C_1 after step T.

Definition 6.1. The PCREW PRAM M computes $f \in \mathbf{B}_n$ if

$$\text{Prob}(M \text{ outputs } f(\mathbf{a}) \text{ on input } \mathbf{a}) > \tfrac{1}{2}$$

for all inputs $\mathbf{a}$. Moreover, M computes f with (two-sided) bounded error ε, $0 < \varepsilon < 1$, if

$$\text{Prob}(M \text{ outputs } f(\mathbf{a}) \text{ on input } \mathbf{a}) > \tfrac{1}{2}(1 + \varepsilon)$$

for all inputs $\mathbf{a}$. If ε is not specified, we talk about an unbounded error computation. (This is useful when considering a family of functions, e.g., PARITY_n for $n \in \mathbf{N}$.)

Before we embark on formulating and proving the results of this section, we must deal with a technical problem. We would like to replace the many random experiments performed by the processors of the PCREW PRAM by only one experiment performed right at the beginning of the computation. However, these random experiments might generate a huge probability space; possibly infinitely many processors are choosing one out of possibly infinitely many new states; the total number of possible computations might thus even be uncountable. This huge space can be reduced to a much smaller, even finite, space by considering the *functions* computed by these computations instead of the computations themselves. The details are given in the following lemma. It justifies the following simpler way of defining probabilistic CREW PRAM computations: In the first step, processor P_1 chooses a (finite) program for computing a function $g \in \mathbf{B}_n$, according to some distribution on $\mathbf{B}_n$, and writes this program to some fixed cell for all other processors to read. In the following steps, all processors compute deterministically according to the program chosen. (See Lemma 5.1 for the corresponding fact for nondeterministic PRAMs.)

LEMMA 6.2. *Assume that $f \in \mathbf{B}_n$ is computed by some PCREW PRAM M (with bounded or unbounded error) in T steps. For $g \in \mathbf{B}_n$, let*

$$p_g := \text{Prob}(\forall_{\mathbf{a} \in \{0,1\}^n} M \text{ outputs } g(\mathbf{a}) \text{ on input } \mathbf{a}).$$

Then

(a) $\sum_{g \in \mathbf{B}_n} p_g = 1$, *and*

(b) $p_g > 0$ *implies* $\text{CREW}(g) \leqslant T$, *for all* $g \in \mathbf{B}_n$.

(*Thus, the family* $\{p_g := g \in \mathbf{B}_n\}$ *defines a probability measure on the set* $\{g \in \mathbf{B}_n : \text{CREW}(g) \leqslant T\}$.)

Proof. Recall the assumption that the random experiment performed by processor P_i in step t depends only on the state q of the processor and on the symbol s it read in step t, but it is independent of the experiments of the other processors and of the previous experiments of P_i. This assumption leads to an alternative description of the random process that governs the operation of M. We simply perform all possible random experiments before the computation starts and use the results to direct a *deterministic* computation of M. (Many of the experiments will not be used at all in this computation.) Formally, let distributions $\mu_{q,s}$ on Q be given, where $\mu_{q,s}$ governs the random experiment of P_i when it is in state q and reads symbol s. Consider the product probability measure

$$\text{Prob}_0 := \prod_{(q,s) \in Q \times \Sigma} \mu_{q,s}$$

on the space $Q^{Q \times \Sigma}$ of all transition functions $\delta: Q \times \Sigma \to Q$ of M. Each such transition function defines a PRAM M_δ. Obviously, we have

$$\text{Prob}_0(\text{the computation of } M_\delta \text{ on input } \mathbf{a} \text{ has property } E)$$

$$= \text{Prob}(\text{the computation of } M \text{ on input } \mathbf{a} \text{ has property } E)$$

for all "reasonable" (i.e., measurable) properties E. Many of the M_δ will produce write conflicts on some or all inputs. However, since we have assumed that for each fixed input $\mathbf{a} \in \{0,1\}^n$ the PRAM M will produce write conflicts only with probability 0, the event

$$\text{non-EW} := \{\delta : \exists_{\mathbf{a} \in \{0,1\}^n} M_\delta \text{ produces a write conflict on input } \mathbf{a}\}$$

is a union of 2^n events of probability zero; hence it has probability zero itself. Thus, if we define $\Delta := Q^{Q \times \Sigma} - \text{non-EW}$ and restrict Prob_0 to Δ so as to obtain a probability measure Prob_Δ, then Prob_Δ still describes the behaviour of M. Moreover, all M_δ for $\delta \in \Delta$ have the exclusive-write property. Clearly, each M_δ

computes some $g_\delta \in \mathbf{B}_n$ with $\mathrm{CREW}(g_\delta) \leqslant T$; further, for p_g as in the formulation of the lemma, we have

$$\begin{aligned} p_g &= \mathrm{Prob}(\forall_{\mathbf{a} \in \{0,1\}^n} M \text{ outputs } g(\mathbf{a}) \text{ on input } \mathbf{a}) \\ &= \mathrm{Prob}_\Delta(\forall_{\mathbf{a} \in \{0,1\}^n} M_\delta \text{ outputs } g(\mathbf{a}) \text{ on input } \mathbf{a}) \\ &= \mathrm{Prob}_\Delta(M_\delta \text{ computes } g) \\ &= \mathrm{Prob}_\Delta(g_\delta = g). \end{aligned}$$

Assertion (a) of the lemma is now clear, and (b) follows since $p_g > 0$ implies that $g = g_\delta$ for some $\delta \in \Delta$; hence $\mathrm{CREW}(g) = \mathrm{CREW}(g_\delta) \leqslant T$. ∎

COROLLARY 6.3. *If $f \in \mathbf{B}_n$ is computed in T steps by a PCREW PRAM with bounded error, then*

(a) *f is computed by a probabilistic decision tree of depth 2^{8T} (with bounded error);*

(b) $T \geqslant \frac{1}{24} \cdot \log(\deg(f)) - O(1)$.

Proof. (a) We use the most general model of probabilistic decision trees, as described, e.g., in [10]. Essentially, given the input, some decision tree that obeys the depth bound is chosen according to some fixed distribution, and then this decision tree is applied to the input. With this model, the transition from a probabilistic CREW PRAM to a probabilistic decision tree is obvious, via Nisan's result $\mathrm{CREW}(f) \geqslant \log(D(f)^{1/8})$, and Lemma 6.2.

(b) In [18, Theorem 4] it is shown that $D_2(f) \geqslant (D(f)/8)^{1/3}$, where $D_2(f)$ is the depth of a probabilistic decision tree (with two-sided bounded error) that computes f. The claimed inequality follows by combining this with the inequalities $2^{8T} \geqslant D_2(f)$ (see part (a)) and $D(f) \geqslant \deg(f)$ (see Fact 2.4). ∎

Remark. We note that there are Boolean functions whose probabilistic (bounded error) CREW-complexity is smaller by a constant factor than the deterministic CREW-complexity. This is an immediate consequence of the existence of functions whose randomized decision tree complexity is $n^{1-\varepsilon}$ while the deterministic decision tree complexity is n. As an example, we can take the following family of functions, which was considered in [25]:

$$g_1(a_1, a_2) = \mathrm{NAND}(a_1, a_2) = 1 - a_1 a_2,$$

for $a_2, a_2 \in \{0, 1\}$; and, for $h > 1$,

$$g_h(a_1, \ldots, a_{2^h}) = g_1(g_{h-1}(a_1, \ldots, a_{2^{h-1}}), g_{h-1}(a_{2^{h-1}+1}, \ldots, a_{2^h})),$$

for $a_1, \ldots, a_{2^h} \in \{0, 1\}$. Let $n = 2^h$ for some h. It is easily seen (by induction) that $\deg(g_h) = n$, hence, by Theorem 1.10, that $\mathrm{CREW}(g_h) \geqslant \phi(n)$. On the other hand, it is shown in [25] that g_h has randomized decision tree complexity $D \leqslant e \cdot n^{0.753\ldots}$ for

some constant e. This bound concerns the expected number of variables tested in the randomized decision tree (which always gives the correct answer). The probability that the decision is made after more than $3D$ steps is at most $\frac{1}{3}$, on each input. Hence, if we cut off each decision tree computation after step $3D$ and answer 0, then the error probability is at most $\frac{1}{3}$, on each input. The resulting randomized decision tree can be regarded as given by a probability distribution on all decision trees of depth at most $3D$. If we replace each such decision tree by a CREW algorithm that computes the same function and takes $\phi(3D)+1$ steps, we obtain a PCREW algorithm with complexity $\phi(3D)+1 \approx 73 \cdot 0.75 \cdot \log n + O(1) \approx 0.55 \log n$, which is smaller than CREW(g_n) by a factor of 0.75....

In the following, we considerably sharpen the inequality given in part (b) of the corollary. In certain cases (PARITY$_n$ being the prime example) we obtain that randomization does not help at all, not even in the unbounded error model. In general, we combine Theorem 3.1 with Fact 2.6 to obtain a much tighter lower bound in the bounded error model.

LEMMA 6.4. *If f is computed by some PCREW PRAM M in T steps (bounded or unbounded error), then there is a real-valued functions $h \in \mathbf{R}^{\{0,1\}^n}$ with* $\deg(h) \leqslant F_{2T+1}$, *so that* $0 \leqslant h(\mathbf{a}) \leqslant 1$ *for all* $\mathbf{a} \in \{0,1\}^n$ *and*

$$\text{Prob}(M \text{ outputs } \overline{f(\mathbf{a})} \text{ on input } \mathbf{a}) = |h(\mathbf{a}) - f(\mathbf{a})|, \qquad \text{for all } \mathbf{a} \in \{0,1\}^n.$$

Proof. Let the probabilities p_g, $g \in \mathbf{B}_n$, be as in Lemma 6.2. If $p_g > 0$, then CREW$(g) \leqslant T$; thus $\deg(g) \leqslant F_{2T+1}$, by Theorem 3.1. Define the real polynomial h as $\sum_{g \in \mathbf{B}_n} p_g \cdot g$. Clearly, $\deg(h) \leqslant F_{2T+1}$. Let $\mathbf{a} \in \{0,1\}^n$. If $f(\mathbf{a}) = 1$, then $|f(\mathbf{a}) - h(\mathbf{a})| = 1 - h(\mathbf{a}) = \sum_{g \in \mathbf{B}_n} p_g \cdot (1 - g(\mathbf{a}))$ is the probability that M gives the answer 0; if $f(\mathbf{a}) = 0$, then $|h(\mathbf{a}) - f(\mathbf{a})| = h(\mathbf{a}) = \sum_{g \in \mathbf{B}_n} p_g \cdot g(\mathbf{a})$ is the probability that M gives the answer 1. ∎

THEOREM 6.5. *If a PCREW PRAM computes* PARITY$_n$ *in T steps (with unbounded error), then $T \geqslant \phi(n)$.*

Proof. Suppose otherwise. Then $r = \deg(h) \leqslant F_{2T+1} < n$ for h as in Lemma 6.4. By (2) in the proof of Fact 2.1, applied to $S = \{1, \ldots, n\}$, we obtain $0 = \alpha_{\{1,\ldots,n\}}(h) = (-1)^n \cdot \sum_{\mathbf{a} \in \{0,1\}^n} h(\mathbf{a}) \cdot (-1)^{\text{PARITY}_n(\mathbf{a})}$; hence

$$\sum_{\mathbf{a} \in \text{PARITY}_n^{-1}(0)} h(\mathbf{a}) = \sum_{\mathbf{a} \in \text{PARITY}_n^{-1}(1)} h(\mathbf{a}).$$

Thus, it cannot be the case that $h(\mathbf{a}) < \frac{1}{2}$ for all $\mathbf{a}$ of even parity and $h(\mathbf{a}) > \frac{1}{2}$ for all $\mathbf{a}$ of odd parity, contradicting Lemma 6.4. ∎

THEOREM 6.6. (a) *If a Boolean function f is computed by a PCREW PRAM in T steps with bounded error ε, then $T \geqslant \phi(\sqrt{\varepsilon \cdot \text{bc}(f)})$.*

(b) *The PCREW time complexity of a Boolean function in the bounded error model differs from the deterministic CREW complexity at most by a factor of* 8.

Proof. (a) In the case of bounded error, the function h from Lemma 6.4 even satisfies $|f(\mathbf{a})-h(\mathbf{a})| \leqslant \frac{1}{2}(1-\varepsilon)$ for all $\mathbf{a} \in \{0,1\}^n$. Thus we may apply Fact 2.6 to conclude that $\deg(h) \geqslant \sqrt{\varepsilon \cdot \mathrm{bc}(f)}$. But $\deg(h) \leqslant F_{2T+1}$; hence we obtain $\phi(\sqrt{\varepsilon \cdot \mathrm{bc}(f)}) \leqslant T$.

(b) By Theorem 1.4(c) and the inequality $D(f) \leqslant \mathrm{bc}(f)^4$ (which follows from Facts 2.7 and 2.8) we obtain that $\mathrm{CREW}(f) \leqslant \phi((\mathrm{bc}(f))^4)+1$. Together with (a) this yields the result. ∎

Remark. Very recently, Paturi [21] succeeded in characterizing the minimal degree necessary for approximating a symmetric function f by a polynomial. In combination with Lemma 6.4 his results allow us to tighten the lower bound of Theorem 6.6(a) for many symmetric functions. Let $0<\varepsilon<1$ be an arbitrary constant, and let $f \in \mathbf{B}_n$ be symmetric (cf. Section 4.3). For $0 \leqslant k \leqslant n$, define f_k to be 1 if $f(\mathbf{a})=1$ for all $\mathbf{a}$ with $\|\mathbf{a}\|=k$, and 0 otherwise. Then let

$$\gamma(f) := \max\{k : 1 \leqslant k \leqslant \tfrac{1}{2}n \text{ and } f_k \neq f_{k+1} \text{ or } f_{n-k+1} \neq f_{n-k}\}$$

(this measures the maximum Hamming distance from both "poles" (0, ..., 0) and (1, ..., 1) at which f switches its value). The lower bound part of Paturi's result reads as follows: If $h: \{0,1\}^n \to \mathbf{R}$ satisfies $|f(\mathbf{a})-h(\mathbf{a})| \leqslant \frac{1}{2}(1-\varepsilon)$ for all $\mathbf{a} \in \{0,1\}^n$, then $\deg(h) = \Omega(\sqrt{n\gamma(f)})$. Exactly as in the proof of Theorem 6.6(a) we conclude that if f is computed by a PCREW PRAM in time T with bounded error ε, then $T \geqslant \phi(\sqrt{n\gamma(f)}) - O(1)$. Many important symmetric functions f satsify $\gamma(f) = \Omega(n)$ (e.g., MAJORITY and MOD_m for $m \leqslant \frac{1}{2}n$); hence they have PCREW complexity $\phi(n) - O(1)$, which differs from $\mathrm{CREW}(f)$ only by an additive constant.

7. Concurrent-Read "Few-Write" PRAMs

In this section, we consider the PRAM model in which concurrent writes to the same cell are permitted. Write conflicts are resolved by the *priority rule*; i.e., in the case of a write conflict the lowest numbered processor wins. As is common, we call such a machine a PRIORITY PRAM. We are interested in the case where only a "few" processors are allowed to simultaneously write to any given memory cell, say $\rho = \rho(n)$ many. It is easily seen that such machines can compute OR_n in $O(\log n/\log \rho(n))$ steps, with n processors; hence any Boolean function can be computed within this time bound, if arbitrarily many processors are available. We sketch the proof of a matching lower bound, thus generalizing Theorems 1.6 and 3.1. (J. Hromkovič proved this lower bound for *oblivious* computations.) Applying the techniques used in [7, proof of Theorem 6] (also see [9]), one can simulate

"few-write" machines on CREW PRAMs with a slowdown $O(\rho(n))$, thus proving the lower bound $\Omega(\log(\deg(f))/\rho(n))$ for $f \in \mathbf{B}_n$. No better bounds are available so far; the results in [9, Theorem 6] imply that none can be achieved by simulation techniques.

THEOREM 7.1. *Let $f \in \mathbf{B}_n$ be computed in T steps by a PRIORITY PRAM M in whose computations at most $\rho = \rho(n)$ processors write into one cell in any step. Then $T \geqslant \log(\deg(f))/\log(\rho + 2) - O(1)$.*

Proof (Sketch). As before, we may assume that M is a "full information" machine. We proceed as in the proof of Lemma 3.2 and show:

(a) $\deg(\mathscr{G}(0)) = 0$ and $\deg(\mathscr{H}(0)) = 1$.

(b) $\deg(\mathscr{G}(t)) \leqslant \deg(\mathscr{H}(t-1)) + \deg(\mathscr{G}(t-1))$, for $0 < t \leqslant T$.

(c) $\deg(\mathscr{H}(t)) \leqslant \deg(\mathscr{H}(t-1)) + \rho \cdot \deg(\mathscr{G}(t))$, for $0 < t \leqslant T$.

We let $b(\rho) := 1 + \rho/2 + ((1 + \rho/2)^2 - 1)^{1/2} \leqslant \rho + 2$, for $\rho \geqslant 1$, and further show that

(d) $\deg(\mathscr{H}(t)) = O(b(\rho)^t)$, for $0 \leqslant t \leqslant T$.

First, we show that the theorem follows from (d): Since M computes f, we have $\deg(\mathscr{H}(T)) \geqslant \deg(f)$; further, by (d), we have $\deg(\mathscr{H}(T)) = O(b(\rho)^T)$. Thus, $T \geqslant \log_{b(\rho)}(\deg(f)) - O(1) = \log(\deg(f))/\log(b(\rho)) - O(1)$.

Next, we indicate how (d) can be derived from (a)–(c): Consider the recurrence relation $g_0 = 0$, $h_0 = 1$, $g_t = h_{t-1} + g_{t-1}$, $h_t = h_{t-1} + \rho g_t$, for $t \geqslant 1$. These equations imply that $h_1 = \rho + 1$ and $h_t = (\rho + 2)\, h_{t-1} - h_{t-2}$, for $t \geqslant 2$. Standard methods for solving linear recurrences yield the solution $h_t = \Theta(b(\rho)^t)$ for $t \geqslant 1$. On the other hand, (a)–(c) imply by an easy induction that $\deg(\mathscr{H}(t)) \leqslant h_t$ for $t \geqslant 1$. This proves (d).

Parts (a), (b), and (c) are proved simultaneously by induction on t. Since parts (a) and (b) are exactly the same as in Lemma 3.2, we only describe how to prove (c). Let $H' = H(s, j, t) \neq \varnothing$ for some $s \in \Sigma$, $j \geqslant 1$, and let $\mathbf{a} \in H'$. Let $\mathscr{G}_j \subseteq \mathscr{G}(t)$ be the set of all (nonempty) classes $G(q, i, t)$ so that P_i writes into C_j when in state q.

Case 1. $\mathbf{a} \in G$ *for some* $G \in \mathscr{G}_j$. Let P_{i_0} be the processor that writes into C_j on input $\mathbf{a}$; i.e., i_0 is minimal so that $\mathbf{a} \in G(q_0, i_0, t) \in \mathscr{G}_j$ for some q_0. Let $\mathscr{G}_j' := \{G(q, i, t) \in \mathscr{G}_j : i < i_0\}$. Then it is not hard to show that $H' = G(q_0, i_0, t) \cap \bigcap \{\bar{G} : G \in \mathscr{G}_j'\}$. Now, by the familiar inclusion–exclusion formula,

$$\chi\left(\bigcap \{\bar{G} : G \in \mathscr{G}_j'\}\right) = \sum (-1)^{|\mathscr{A}|} \cdot \prod_{G \in \mathscr{A}} \chi_G,$$

where the sum extends over sets $\mathscr{A} \subseteq \mathscr{G}_j'$; further, the "few-write" rule implies that for all $\mathbf{b} \in G(q_0, i_0, t)$ at most $\rho - 1$ many $G \in \mathscr{G}_j$ contain $\mathbf{b}$, that means $\chi(G(q_0, i_0, t)) \cdot \chi(\prod_{G \in \mathscr{A}} \chi_G) \equiv 0$ if $|\mathscr{A}| \geqslant \rho$. Hence, $\deg(\chi_{H'}) \leqslant \deg(\mathscr{G}(t)) + (\rho - 1) \cdot \deg(\mathscr{G}(t))$.

Case 2. $\mathbf{a} \notin G$ *for all* $G \in \mathscr{G}_j$. Then $H' = H(s, j, t-1) \cap \bigcap \{\bar{G}: G \in \mathscr{G}_j\}$. Arguing essentially as in Case 1, we obtain $\deg(\chi_{H'}) \leqslant \deg(\chi(H(s, j, t-1))) + \rho \cdot \deg(\mathscr{G}(t))$, which suffices to prove (c). ∎

Remark. It is not very hard to show that an arbitrary Boolean function f can be computed on a "few-write" PRAM as above in time $\log(D(f))/\log(b(\rho)) + O(1)$, which matches the lower bound of Theorem 7.1 up to a constant factor (cf. Corollary 2.9), in case $D(f) = \deg(f)$ even up to a constant additive term.

We sketch the proof of this upper bound. First note that it is sufficient to show that OR_n can be computed in $\log(n)/\log(b(\rho)) + O(1)$ steps, cf. the proof of Theorem 1.4(c). We let g_t, h_t, $t \geqslant 0$, have the same meaning as in the proof of Theorem 7.1. Mimicking the construction of [7] for computing OR_n in $\phi(n)$ steps on a CREW PRAM we can arrange that on the "few-write" PRAM after t steps each processor knows the OR of a group of g_t input bits and each memory cell stores the value of the OR of a group of h_t bits. Then, clearly, $\min\{t: h_t \geqslant n\}$ steps are sufficient to compute OR_n. As $h_t = \Theta(b(\rho)^t)$, the claimed upper bound follows.

Acknowledgments

The first author thanks Friedhelm Meyer auf der Heide for many discussions in the course of the development of the lower bound argument; thanks to Ilan Newman for pointing out Szegedy's results; Juraj Hromkovič brought up the idea of "few-write" machines and proved the lower bound for the oblivious case; thanks also to Paul Beame and Noam Nisan for allowing us to include Example 2.11 in this paper. We also thank two anonymous referees for their thorough reading of the paper and their comments that helped to improve the presentation of our results.

References

1. P. Beame, Limits on the power of concurrent-write parallel machines, *Inform. and Comput.* **76** (1988), 13–28.
2. P. Beame and N. Nisan, personal communication.
3. M. Blum and R. Impagliazzo, Generic oracles and oracle classes, *in* "Proceedings, 28th IEEE Sympos. Found. of Computer Sci., 1987," pp. 118–126.
4. J. Bruck, Harmonic analysis of polynomial threshold functions, *SIAM J. Discrete Math.* **3** (1990), 168–177.
5. J. Bruck and R. Smolensky, Polynomial threshold functions, AC^0 functions, and spectral norms, *in* "Proceedings, 31st IEEE Sympos. Found. of Computer Sci., 1990," pp. 632–641.
6. S. Bublitz, U. Schürfeld, B. Voigt, and I. Wegener, Properties of complexity measures for PRAMs and WRAMs, *Theoret. Comput. Sci.* **48** (1986), 53–73.
7. S. Cook, C. Dwork, and R. Reischuk, Upper and lower time bounds for parallel random access machines without simultaneous writes, *SIAM J. Comput.* **15** (1986), 87–97.
8. M. Dietzfelbinger, M. Kutyłowski, and R. Reischuk, "Realistic Time-Optimal Algorithms for Boolean Functions on Exclusive-Write PRAMs," Technical Report 96, Fachbereich Mathematik-Informatik, Universität-Gesamthochschule-Paderborn, Germany, December 1991. (Journal version to appear in *SIAM J. Comput.*)
9. T. Hagerup and M. Nowak, Parallel retrieval of scattered information, *in* "Proceedings, 16th International Colloquium on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 372, pp. 439–450, Springer-Verlag, Berlin, 1989.

10. P. HAJNAL, On the power of randomness in the decision tree model, *in* "Proceedings, 5th Structure in Complexity Theory Conference, 1990," pp. 66–77.
11. J. HARTMANIS AND L. A. HEMACHANDRA, "One-Way Functions, Robustness, and Nonisomorphism of NP-Complete Sets," Report DCS TR86-796, Dept. of Computer Science, Cornell University, Ithaca, NY, 1987.
12. J. KAHN, G. KALAI, AND N. LINIAL, The influence of variables on boolean functions, *in* "Proceedings, 29th IEEE Sympos. Found. of Computer Sci., 1988," pp. 68–80.
13. M. KUTYŁOWSKI, Time complexity of Boolean functions on CREW PRAMs, *SIAM J. Comput.* **20** (1991), 824–833.
14. M. KUTYŁOWSKI AND R. REISCHUK, "Evaluating Formulas on Parallel Machines without Simultaneous Writes," Technical Report, Institut für Theoretische Informatik, Technische Hochschule Darmstadt, Germany, January 1990.
15. R. J. LECHNER, Harmonic analysis of switching functions, *in* "Recent Developments in Switching Theory" (A. Mukhopadhyay, Ed.), pp. 121–228, Academic Press, New York, 1971.
16. N. LINIAL, Y. MANSOUR, AND N. NISAN, Constant depth circuits, Fourier transform, and learnability, *in* "Proceedings, 30th IEEE Sympos. Found. of Computer Sci., 1989," pp. 574–579.
17. M. MINSKY AND S. PAPERT, "Perceptrons," MIT Press, Cambridge, MA, 1969.
18. N. NISAN, CREW PRAMs and decision trees, *SIAM J. Comput.* **20** (1991), 999–1007.
19. N. NISAN AND M. SZEGEDY, On the degree of boolean functions as real polynomials, *in* "Proceedings, 24th ACM Sympos. Theory of Computing, 1992," pp. 462–467.
20. I. PARBERRY AND P. Y. YAN, Improved upper and lower time bounds for parallel random access machines without simultaneous writes, *SIAM J. Comput.* **20** (1991), 88–99.
21. R. PATURI, On the degree of polynomials that approximate symmetric boolean functions, *in* "Proceedings, 24th ACM Sympos. Theory of Computing, 1992," pp. 468–474.
22. A. A. RAZBOROV, Lower bounds on the size of bounded depth networks over a complete basis with logical addition, *Mat. Zametki* **41**, No. 4 (1987), 598–607; English transl., *Math. Notes* **41**, No. 4 (1987), 333–338.
23. I. S. REED, A class of multiple error-correcting codes and the decoding scheme, *IRE Trans. Inform. Theory* **IT-4** (1954), 38–49.
24. R. L. RIVEST AND J. VUILLEMIN, On recognizing graph properties from adjacency matrices, *Theoret. Comput. Sci.* **3** (1976), 371–384.
25. M. SAKS AND A. WIGDERSON, Probabilistic boolean decision trees and the complexity of evaluating game trees, *in* "Proceedings, 27th IEEE Sympos. Found. of Computer Sci., 1986," 29–38.
26. H. U. SIMON, A tight $\Omega(\log \log n)$ bound on the time for parallel RAM's to compute nondegenerate Boolean functions, *Inform. and Control* **55** (1982), 102–107.
27. R. SMOLENSKY, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, *in* "Proceedings, 19th ACM Sympos. Theory of Computing, 1987," pp. 77–82.
28. M. SNIR, On parallel searching, *SIAM J. Comput.* **14** (1985), 688–708.
29. M. SZEGEDY, "Algebraic Methods in Lower Bounds for Computational Models with Limited Communication," Ph.D. dissertation, University of Chicago, 1989.
30. G. TARDOS, Query complexity, or why is it difficult to separate $NP^A \cap co\text{-}NP^A$ from P^A by a random oracle A?, *Combinatorica* **9** (1989), 385–392.
31. I. WEGENER, "The Complexity of Boolean Functions," Wiley-Teubner, Stuttgart, 1987.